Poor Man's Parallelism

Background

Fractals are mathematical sets, typically defined by an iterative function. They can produce very visually appealing pictures, which exhibits self-similarity. Due to their iterative nature, the rendering of these fractals can be very time-consuming. At the same time, the work is quite easy to parallelize.

Functionality

Your task is to create two applications, which together renders parts of the Mandelbrot set [1]. One is a server that should accept requests on a TCP port. The other is a client, which should spread the workload over a set of servers.

Interface

The server should accept a request that has this form, or something similar:

GET /mandelbrot/{min_c_re}/{min_c_im}/{max_c_re}/{max_c_im}/{x}/{y}/{inf_n}

In return, it should return a gray-scale image of dimension x times y, where each pixel corresponds to the number of iterations it takes, until $|z_n| > c$, $z_{n+1} = z_n^2 + c$, (c is complex), with the pixel value is modulo 256, and a maximum on inf_n iterations is performed. c is varied over the image between the boundaries given. The format of the image can either be simply a set of bytes, one per pixel, or for example a PGM image[2] (a set of bytes, one per pixel, with a header).

The server should be capable of handle multiple calls, and it should be possible to specify which port the server listens to.

The client is a command-line tool with these arguments:

min_c_re min_c_im max_c_re max_c_im max_n x y divisions list-of-servers

Example: -1 -1.5 2 1.5 1024 10000 10000 4 localhost:4444 localhost:3333 192.168.33.3:4444

This should divide the 10000x10000 picture in 4x4 sub-pictures, and spread the work to the three given servers. The output from the client should be a PGM image or other image format.

(Note that some parts of the picture will be significantly faster to render than others.)

Implementation

You may choose freely among programming languages and tools used. However, make sure your choices expose your capabilities and expertise. (And if you choose a bizarre programming language, we might have doubts about the maintainability of the code you produce :-))

Criteria and Documentation

There is no Pass or Fail. Even if your code does not work 100%, we are still very interested to take a look at it.

One final note. Documentation is the Alpha and the Omega.

[1] http://en.wikipedia.org/wiki/Mandelbrot_set
[2] http://en.wikipedia.org/wiki/Portable_graymap

© Vidispine AB 2015